Recurrent Interpolants for Probabilistic Time Series Prediction

Yu Chen [1 *] **Marin Biloš** [1*]

Sarthak Mittal [2*] **Wei Deng** [1] **Kashif Rasul** [1] **Anderson Schneider** [1]

Abstract

Sequential models such as recurrent neural networks or transformer-based models became *de facto* tools for multivariate time series forecasting in a probabilistic fashion, with applications to a wide range of datasets, such as finance, biology, medicine, etc. Despite their adeptness in capturing dependencies, assessing prediction uncertainty, and efficiency in training, challenges emerge in modeling high-dimensional complex distributions and cross-feature dependencies. To tackle these issues, recent works delve into generative modeling by employing diffusion or flow-based models. Notably, the integration of stochastic differential equations or probability flow successfully extends these methods to probabilistic time series imputation and forecasting. However, scalability issues necessitate a computational-friendly framework for large-scale generative model-based predictions. This work proposes a novel approach by blending the computational efficiency of recurrent neural networks with the high-quality probabilistic modeling of the diffusion model, which addresses challenges and advances generative models' application in time series forecasting. Our method relies on the foundation of stochastic interpolants and the extension to a broader conditional generation framework with additional control features, offering insights for future developments in this dynamic field.

1 Introduction

Autoregression models [Box et al., 2015], such as recurrent neural networks [Graves, 2013, Sutskever et al., 2014, Hochreiter and Schmidhuber, 1997] or transformer models [Vaswani et al., 2017], have been the go-to methods for time series forecasting in various datasets [Morrill et al., 2021]. These methods can also provide an assessment of prediction uncertainty through probabilistic forecasting by incorporating specific parametric probabilistic models into the output layer of the neural network. However, the probabilistic output layer is confined within a simple probability family because the density needs to be parameterized by neural networks, and the loss must be differentiable with respect to neural network parameters.

To better capture sophisticated distributions in time series modeling and learn both the temporal and cross-feature dependencies, a common strategy involves exploring the generative modeling of time series using efficient distribution transportation plans, especially via diffusion or flow based models. For example, recent works such as Li et al. [2020] propose using latent neural SDE as latent state for modeling time series in a stochastic manner, while Spantini et al. [2022] summarize non-linear extensions of state space models using both deterministic and stochastic transformation plans. Tashiro et al. [2021], Biloš et al. [2023], Chen et al. [2023], Miguel et al. [2022], Li et al. [2022], Deng et al. [2024a,b], Chen et al. [2024] studied the application of diffusion models in probabilistic time series

*Equal Contributions. [1] Machine Learning Research, Morgan Stanley, [2] Mila, Université de Montréal. Mittal completed part of the work while interning at Morgan Stanley. Correspondence: weideng056@gmail.com

NeurIPS 2024 Third Table Representation Learning Workshop .

imputation and forecasting. Compared to recurrent model, a more computational friendly framework is needed for large scale generative model-based time series prediction problems.

These observations inspire the creation of a time series prediction model under the generative framework that maps between dependent data points: Initiating the prediction of future time point's distribution with the current time point is more straightforward and yields better quality; meanwhile, the longer temporal dependency is encoded by a recurrent neural network and the embedded history is passed to the generative model as the guidance of the prediction for the future time points. The new framework benefits from the efficient training and computation inherited from the recurrent neural network, while enjoying the high quality of probabilistic modeling empowered by the diffusion model. In this paper, we first extend the theory of stochastic interpolants to more general conditional generation framework with extra control features. Then we describe the proposed method, which we adopt a conditional stochastic interpolants module for the sequential modeling and time series prediction, which is computational-friendly and achieves high quality modeling of the future time point's distribution.

2 Stochastic Interpolants for Time Series Prediction

We formulate time series prediction tasks through the conditional probability $p(\mathbf{x}_{t+1}|\mathbf{x}_{t-P:t})$. The model diagram is illustrated in Figure 1. Here, $\mathbf{x}_t \in \mathbb{R}^D$ represents the multivariate time series at time t with D dimensions, $\mathbf{x}_{t+1}$ is the prediction target, and $\mathbf{x}_{t-P:t}$ is the context window, where P denotes the length of the context window.

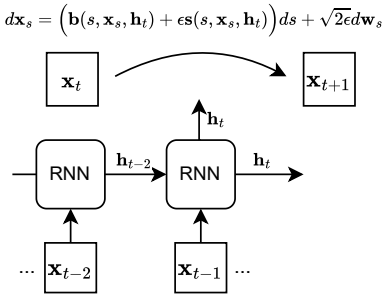

Figure 1: Model architecture for recurrent interpolant models.

For this problem, we employ the conditional Stochastic Interpolants (SI) method as follows. In the *training phase*, the generative model learns the joint distribution $p(\mathbf{x}_{t+1}, \mathbf{x}_t|\mathbf{x}_{t-P:t-1})$ of the pair $(\mathbf{x}_{t+1}, \mathbf{x}_t)$ given the past observations $\mathbf{x}_{t-P:t-1}$. where $\mathbf{x}_t \sim \rho_0$ and $\mathbf{x}_{t+1} \sim \rho_1$ for all t, so the marginal distributions are equal $\rho_0 = \rho_1$. The model aims to learn the coupling relation between $\mathbf{x}_{t+1}$ and $\mathbf{x}_t$ conditioning on the context $\mathbf{x}_{t-P:t-1}$. This is achieved by training the conditional velocity and score functions in equation 1.

As the sample spaces of ρ_0 and ρ_1 must be the same, the generative model can not directly map the whole context window $\mathbf{x}_{t-P:t}$ to the target $\mathbf{x}_{t+1}$ due to different vector sizes. Instead, a recurrent neural network is used to encode the context $\mathbf{x}_{t-P:t-1}$ into a *history prompt* $\mathbf{h}_t$. Subsequently, the score function and velocity function perform conditional generation diffusing from $\mathbf{x}_t$ with the condition input $\mathbf{h}_t$.

2.1 Training of Conditional Stochastic Interpolant

Regarding the conditional stochastic interpolants, the inference using forward or backward SDEs are the following:

$$\mathrm{d}\mathbf{x}_s = [\mathbf{b}(s, \mathbf{x}_s, \xi) + \epsilon(s)\mathbf{s}(s, \mathbf{x}_s, \xi)]\mathrm{d}s + \sqrt{2\epsilon(s)}\mathrm{d}\mathbf{w}_s \tag{1}$$

$$\mathrm{d}\mathbf{x}_s = [\mathbf{b}(s, \mathbf{x}_s, \xi) - \epsilon(s)\mathbf{s}(s, \mathbf{x}_s, \xi)]\mathrm{d}s + \sqrt{2\epsilon(s)}\mathrm{d}\mathbf{w}_s^{\mathrm{B}} \tag{2}$$

where both velocity and score functions depend on the condition ξ.

The SI model is trained to match the equations in equation 8 and equation 9 by minimizing the mean squared error loss functions,

$$\mathcal{L}_b = \int_0^1 \mathbb{E}\Big[\frac{1}{2}\|\hat{\mathbf{b}}(s, \mathbf{x}_s)\|^2 - \big(\dot{\alpha}(s)\mathbf{x}_0 + \dot{\beta}(s)\mathbf{x}_1 + \dot{\gamma}(s)\mathbf{z}\big)^T\hat{\mathbf{b}}(s, \mathbf{x}_s)\Big]\mathrm{d}s \tag{3}$$

$$\mathcal{L}_s = \int_0^1 \mathbb{E}\Big[\frac{1}{2}\|\hat{\mathbf{s}}(s, \mathbf{x}_s)\|^2 + \gamma^{-1}\mathbf{z}^T\hat{\mathbf{s}}(s, \mathbf{x}_s)\Big]\mathrm{d}s \tag{4}$$

Algorithm 1 Training algorithm. i is the sample index.

Input: Sample 3-tuples $(\mathbf{x}_{t_i+1}, \mathbf{x}_{t_i}, \mathbf{x}_{t_i-P:t_i-1})$. Interpolant schedules: $\alpha(s), \beta(s), \gamma(s)$. Models: velocity $\hat{\mathbf{b}}$, score $\hat{\mathbf{s}}$, RNN.
for iteration $= 1$ **to** total iterations **do**
$s_i \sim \mathrm{Beta}(0.1, 0.1)$.
$\mathbf{x}_{s,i} = \alpha(s_i)\mathbf{x}_{t_i} + \beta(s_i)\mathbf{x}_{t_i+1} + \gamma(s_i)\mathbf{z}_i$
$\mathbf{h}_i = \mathrm{RNN}(\mathbf{x}_{t_i-P:t_i-1})$

$$\mathcal{L}_b = \sum_{i=1}^{\text{batch size}} \frac{1}{p_{\mathrm{Beta}}(s_i)} \Big[\frac{1}{2}\|\hat{\mathbf{b}}(s_i, \mathbf{x}_{s,i}, \mathbf{h}_i)\|^2 - \big(\dot{\alpha}(s_i)\mathbf{x}_{t_i} + \dot{\beta}(s_i)\mathbf{x}_{t_i+1} + \dot{\gamma}(s_i)\mathbf{z_i}\big)^T \hat{\mathbf{b}}(s_i, \mathbf{x}_{s,i}, \mathbf{h}_i)\Big]$$

$$\mathcal{L}_s = \sum_{i=1}^{\text{batch size}} \frac{1}{p_{\mathrm{Beta}}(s_i)} \Big[\frac{1}{2}\|\hat{\mathbf{s}}_i(s_i, \mathbf{x}_{s,i})\|^2 + \gamma^{-1}\mathbf{z_i}^T \hat{\mathbf{s}}_i(s_i, \mathbf{x}_{s,i})\Big]$$

Perform back-propagation by minimizing $\mathcal{L}_b$ and $\mathcal{L}_s$.
end for

The training dataset consists of tuple $(\mathbf{x}_{t+1}, \mathbf{x}_t, \mathbf{x}_{t-P:t-1})$. It is worth noting that the loss values become larger when s is close to two ends. To address this, importance sampling is leveraged to better handle the integral over diffusion time in the loss functions equation 3 and equation 4 to stabilize the training, where we use Beta distribution for our proposal distribution. The algorithm is outlined in Algorithm 1. We stabilize the training using importance sampling methods. Demonstration of importance sampling is shown in Appendix C.

2.2 Inference of Conditional Stochastic Interpolant

The RNN first encodes the context $\mathbf{x}_{t-P:t-1}$ into the history prompt $\mathbf{h}_t$, then SI transports the context vector $\mathbf{x}_t$ to the target distribution with the condition $\mathbf{h}_t$, following the forward SDE. Regarding the multiple steps prediction, we recursively run the step-by-step prediction.

Algorithm 2 Inference algorithm

Input: Sample 2-tuples $(\mathbf{x}_t, \mathbf{x}_{t-P:t-1})$. Trained models: Velocity $\hat{\mathbf{b}}$, score $\hat{\mathbf{s}}$, RNN. Diffusion variance ϵ.
Set $\tilde{\mathbf{x}}_0 = \mathbf{x}_t$. $\mathbf{h} = \mathrm{RNN}(\mathbf{x}_{t-P:t-1})$.
Run SDE integral for $s \in [0, 1]$ following

$$\mathrm{d}\tilde{\mathbf{x}}_s = [\mathbf{b}(s, \tilde{\mathbf{x}}_s, \mathbf{h}) + \epsilon\mathbf{s}(s, \tilde{\mathbf{x}}_s, \mathbf{h})]\mathrm{d}s + \sqrt{2\epsilon}\mathrm{d}\mathbf{w}_s$$

Output: $\tilde{\mathbf{x}}_1$ as prediction of $\mathbf{x}_{t+1}$.

3 Experiments

We study our method on four real-world time series forecasting tasks. The time series datasets include: SolarLai et al. [2018], ExchangeLai et al. [2018], Traffic[2], and Wikipedia[3]. We follow the preprocessing steps as in Salinas et al. [2019].

We empirically verify that: 1) SI is a suitable generative module for the prediction compared with other baselines with different generative methods under the same framework; 2) the whole framework can achieve competitive performance in time series forecasting.

Results. The results for CRPS-sum, ND-sum, and NRMSE-sum are shown in Table 1, 3, 4. We outperform or match other models on three out of four datasets, only on Traffic FM model achieves

[2] `https://archive.ics.uci.edu/ml/datasets/PEMS-SF`

[3] `5https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release/datasets`

	Exchange rate	Solar	Traffic	Wiki
Vec-LSTM	0.008±0.001	0.391±0.017	0.087±0.041	0.133±0.002
DDPM	0.009±0.004	**0.359**±0.061	0.058±0.014	0.084±0.023
FM	0.009±0.001	0.419±0.027	**0.038**±0.002	64.256±62.596
SGM	0.008±0.002	0.364±0.029	0.071±0.05	0.108±0.026
SI	**0.007**±0.001	**0.359**±0.06	0.083±0.005	**0.080**±0.007

Table 1: CRPS-sum metric on multivariate probabilistic forecasting. A smaller number indicates better performance.

better performance. Note that on Wiki data FM cannot capture the data distribution. We ran a search over flow matching hyperparameters without being able to get satisfying results. Therefore, we conclude that stochastic interpolants are a strong candidate for conditional generation, in particular for multivariate probabilistic forecasting. By comparing to the RNN-based model Vec-LSTM, our model and other baselines such as SGM and DDPM get better performance, which implies that carefully model the probability distribution is critical for large dimension time series prediction. Figure 2 demonstrates the quality of the forecast on Solar dataset. We can see that our model can make precise prediction and capture the uncertainty, even when the scale of the different dimensions varies a lot.

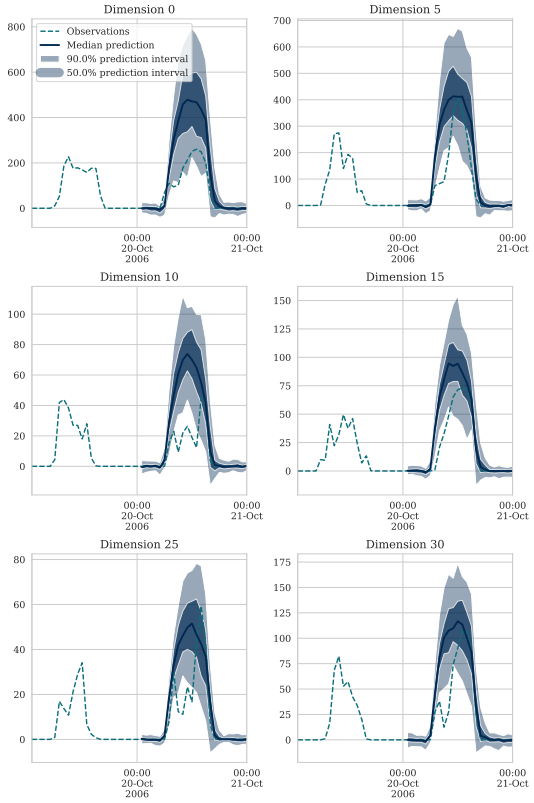

Figure 2: Example forecast paths for SI on Solar dataset. Showing median prediction and confidence intervals calculated from model samples, on 6 out of 137 variate dimensions.

4 Conclusions

This study presents an innovative method that effectively merges the computational efficiency of recurrent neural networks with the high-quality probabilistic modeling of the diffusion model, specifically applied to probabilistic time series forecasting. Grounded in stochastic interpolants and an expanded conditional generation framework featuring control features, the method undergoes empirical evaluation on both synthetic and real datasets, showcasing its compelling performance.

References

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic Interpolants: A Unifying Framework for Flows and Diffusions. In *arXiv:2303.08797v3*, 2023a.

Michael S. Albergo, Mark Goldstein, Nicholas M. Boffi, Rajesh Ranganath, and Eric Vanden-Eijnden. Stochastic Interpolants with Data-Dependent Couplings. *arXiv:2310.03725v2*, 2023b.

Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, et al. Gluonts: Probabilistic time series models in python. *arXiv preprint arXiv:1906.05264*, 2019.

Marin Biloš, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, and Stephan Günnemann. Modeling Temporal Data as Continuous Functions with Process Diffusion. In *Proc. of the International Conference on Machine Learning (ICML)*, 2023.

George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. WILEY, 2015.

Yifan Chen, Mark Goldstein, Mengjian Hua, Michael S. Albergo, Nicholas M. Boff, and Eric Vanden-Eijnden. Probabilistic Forecasting with Stochastic Interpolants and Föllmer Processes. In *Proc. of the International Conference on Machine Learning (ICML)*, 2024.

Yu Chen, Wei Deng, Shikai Fang, Fengpei Li, Nicole Tianjiao Yang, Yikai Zhang, Kashif Rasul, Shandian Zhe, Anderson Schneider, and Yuriy Nevmyvaka. Provably convergent schr\" odinger bridge with applications to probabilistic time series imputation. 2023.

Wei Deng, Yu Chen, Nicole Tianjiao Yang, Hengrong Du, Qi Feng, and Ricky T. Q. Chen. Reflected Schrödinger Bridge for Constrained Generative Modeling. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2024a.

Wei Deng, Weijian Luo, Yixin Tan, Marin Biloš, Yu Chen, Yuriy Nevmyvaka, and Ricky T. Q. Chen. Variational Schrödinger Diffusion Models. In *Proc. of the International Conference on Machine Learning (ICML)*, 2024b.

Alex Graves. Generating Sequences with Recurrent Neural Networks. 2013.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8), 1997.

Alireza Koochali, Peter Schichtel, Andreas Dengel, and Sheraz Ahmed. Random noise vs. state-of-the-art probabilistic forecasting methods: A case study on crps-sum discrimination ability. *Applied Sciences*, 12(10):5104, 2022.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable Gradients for Stochastic Differential Equations. In *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Yan Li, Xinjiang Lu, Yaqing Wan, and Dejing Do. Generative Time Series Forecasting with Diffusion, Denoise, and Disentanglement. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Juan Miguel, Lopez Alcaraz, and Nils Strodthoff. Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models. In *Transactions on Machine Learning Research*, 2022.

James Morrill, Cristopher Salvi, Patrick Kidger, James Foster, and Terry Lyons. Neural Rough Differential Equations for Long Time Series. In *Proc. of the International Conference on Machine Learning (ICML)*, 2021.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Alessio Spantini, Ricardo Baptista, and Youssef Marzouk. Coupling Techniques for Nonlinear Ensemble Filtering. In *SIAM Review*, volume 64:4, page 10.1137, 2022.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *NIPS*, 2014.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A Preliminaries on Stochastic Interpolants (SI)

Stochastic interpolants [Albergo et al., 2023b] aim to model the *dependent couplings* between $(\mathbf{x}_0, \mathbf{x}_1)$ with their joint density $\rho(\mathbf{x}_0, \mathbf{x}_1)$, and establish a two-way generative SDEs mapping from one data distribution to another. The method constructs a straightforward stochastic mapping from $t = 0$ to $t = 1$ given the values at two ends $\mathbf{x}_0 \sim \rho_0$ to $\mathbf{x}_1 \sim \rho_1$, which provides a transport between two densities ρ_0 and ρ_1, while maintaining the dependency between $\mathbf{x}_0$ and $\mathbf{x}_1$.

$$\mathbf{x}_s = \alpha(s)\mathbf{x}_0 + \beta(s)\mathbf{x}_1 + \gamma(s)\mathbf{z},\ s \in [0, 1], \mathbf{z} \sim \mathcal{N}(\mathbf{0}, I) \tag{5}$$

where $\rho(s, \mathbf{x})$ is the marginal density of $\mathbf{x}_s$ at diffusion time s. Such a stochastic mapping is characterized by a pair of functions: velocity function $\mathbf{b}(s, \mathbf{x})$ and score function $\mathbf{s}(s, \mathbf{x})$.

$$\mathbf{s}(s, \mathbf{x}) := \nabla \log \rho(s, \mathbf{x}) \tag{6}$$

$$\mathbf{b}(s, \mathbf{x}) := \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1, \mathbf{z}}[\dot{\alpha}(s)\mathbf{x}_0 + \dot{\beta}(s)\mathbf{x}_1 + \dot{\gamma}(s)\mathbf{z}|\mathbf{x}_s = \mathbf{x}] \tag{7}$$

$\mathbf{b}(s, \mathbf{x})$, $\rho(s, \mathbf{x})$, and $\mathbf{s}(s, \mathbf{x})$ satisfy the equality below,

$$\partial_t \rho(s, \mathbf{x}) + \nabla \cdot (\mathbf{b}(s, \mathbf{x})\rho(s, \mathbf{x})) = 0 \tag{8}$$

$$\mathbf{s}(s, \mathbf{x}) = -\gamma^{-1}(s)\mathbb{E}_{\mathbf{z}}[\mathbf{z}|\mathbf{x}_t = \mathbf{x}] \tag{9}$$

where $\alpha(s)$ and $\beta(s)$ schedule the deterministic interpolant. We set $\alpha(0) = 1, \alpha(1) = 0, \beta(0) = 0, \beta(1) = 1$. $\gamma(s)$ schedules the variance of the stochastic component $\mathbf{z}$. We set $\gamma(0) = \gamma(1) = 0$, so the two ends of the interpolant are fixed at $\mathbf{x}_0$ and $\mathbf{x}_1$. The velocity function $\mathbf{b}(s, \mathbf{x})$ and the score function $\mathbf{s}(s, \mathbf{x})$ can be modeled by a rich family of functions, such as deep neural networks. The model is trained to match the above equality by minimizing the mean squared error loss functions.

During inference, usually, one side of the diffusion trajectory at $t = 0$ or $t = 1$ is given, the goal is to infer the sample distribution on the other side. The interpolant in equation 5 results in elegant forward and backward SDEs and corresponding Fokker-Planck equations, which offer convenient tools for inference. The SDEs are composed of $\mathbf{b}(s, \mathbf{x}_s)$ and $\mathbf{s}(s, \mathbf{x}_s)$, which are learned from the data. For any $\epsilon(s) \geq 0$, define the forward and backward SDEs

$$\mathrm{d}\mathbf{x}_s = [\mathbf{b}(s, \mathbf{x}) + \epsilon(s)\mathbf{s}(s, \mathbf{x})]\mathrm{d}s + \sqrt{2\epsilon(s)}\mathrm{d}\mathbf{w}_s \tag{10}$$

$$\mathrm{d}\mathbf{x}_s = [\mathbf{b}(s, \mathbf{x}) - \epsilon(s)\mathbf{s}(s, \mathbf{x})]\mathrm{d}s + \sqrt{2\epsilon(s)}\mathrm{d}\mathbf{w}^{\mathrm{B}}_s \tag{11}$$

where $\mathbf{w}^{\mathrm{B}}_s$ is the backward Brownian motion. The SDEs satisfy the forward and backward Fokker-Plank equations,

$$\partial_s \rho + \nabla \cdot (\mathbf{b}_{\mathrm{F}}\rho) = \epsilon(s)\Delta\rho, \rho(0) = \rho_0 \tag{12}$$

$$\partial_s \rho + \nabla \cdot (\mathbf{b}_{\mathrm{B}}\rho) = -\epsilon(s)\Delta\rho, \rho(1) = \rho_1 \tag{13}$$

These properties imply that one can draw samples from the conditional intensity $\rho(\mathbf{x}_1|\mathbf{x}_0)$ following the forward SDE in equation 10 starting from $\mathbf{x_0}$ at $s = 0$. It can also draw samples from the joint density $\rho(\mathbf{x}_0, \mathbf{x}_1)$ by initially drawing a sample $\mathbf{x}_0 \sim \rho_0$ (if feasible, for example, pick one sample from the dataset), then using the forward SDE to generate a samples $\mathbf{x}_1$ at $s = 1$. The method guarantees that $\mathbf{x}_1$ follows marginal distribution ρ_1 and the sample pair $(\mathbf{x}_0, \mathbf{x}_1)$ satisfies the joint density $\rho(\mathbf{x}_0, \mathbf{x}_1)$. Drawing samples using the backward SDE is similar: one can draw samples from $\rho(\mathbf{x}_0|\mathbf{x}_1)$ and the joint density $\rho(\mathbf{x}_0, \mathbf{x}_1)$ as well. Details of inference will be shown in section A.

SI can be expanded for conditional generation by substituting the velocity function and score function with $\mathbf{b}(\mathbf{x}_s, s, \xi)$ and $\mathbf{s}(\mathbf{x}_s, s, \xi)$ [Albergo et al., 2023b]. The model is trained using samples of tuples $(\mathbf{x}_0, \mathbf{x}_1, \xi)$, where ξ is the extra condition feature. Theoretical justifications are shown in Appendix B with two Theorems. Regarding the time series prediction task, we will encode a large context window as the conditional information, and the prediction or generation of future time points will rely on such conditional generation mechanism.

B Conditional Stochastic Interpolants

Next, we demonstrate that the probability distribution of x_t as simulated by equation 14, results in a dynamic density function. This density serves as a solution to a transport equation 15, which smoothly transitions between ρ_0 and ρ_1.

Theorem 1 *(Extension of Stochastic Interpolants to Arbitrary Joint Distributions). Let ρ_{01} be the joint distribution $(x_0, x_1) \sim \rho_{01}$ and let the stochastic interpolant be*

$$\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \beta_t \mathbf{x}_1 + \gamma_t \mathbf{z}, \tag{14}$$

where $\alpha_0 = \beta_1 = 1$, $\alpha_1 = \beta_0 = \gamma_0 = \gamma_1 = 0$, and $\alpha_t^2 + \beta_t^2 + \gamma_t^2 > 0$ for all $t \in [0, 1]$. We define ρ_t to be the time-dependent density of $\mathbf{x}_t$, which satisfies the boundary conditions at $t = 0, 1$ and the transport equation follows that

$$\dot{\rho}_t + \nabla \cdot (\mathbf{b}_t \rho_t) = 0 \tag{15}$$

for all $t \in [0, 1]$ with the velocity defined as

$$\mathbf{b}_t(\mathbf{x}|\xi) = \mathbb{E}\left[\dot{\alpha}_t \mathbf{x}_0 + \dot{\beta}_t \mathbf{x}_1 + \dot{\gamma}_t \, \mathbf{z} | \mathbf{x}_t = \mathbf{x}, \xi\right], \tag{16}$$

where the expectation is based on the density ρ_{01} given $\mathbf{x}_s = \mathbf{x}$ and the extra information ξ.

The score function follows the relation such that

$$\nabla \log \rho_t(\mathbf{x}) = -\gamma_t^{-1} \mathbb{E}\left[\mathbf{z} | \mathbf{x}_t = \mathbf{x}, \xi\right].$$

The proof is in a spirit similar to Theorem 2 in [Albergo et al., 2023b]. The key difference is that we consider a continuous-time interpretation and avoid using characteristic functions, which makes the analysis more friendly to users. Additionally, the score function $\nabla \log \rho_t(x)$ is optimized in a simple quadratic objective function.

Proof [Proof of Theorem 1]

Given the conditional information ξ and $\mathrm{x}_s = \mathrm{x}$ simulated from equation 14, the conditional stochastic interpolant for equation 14 follows that

$$\mathbb{E}[\mathrm{x}_t | \mathrm{x}_s = \mathrm{x}, \xi] = \mathbb{E}[\alpha_t \mathrm{x}_0 + \beta_t \mathrm{x}_1 + \gamma_t z | \mathrm{x}_s = \mathrm{x}, \xi], \tag{17}$$

where the expectation takes over the density for $(\mathrm{x}_0, \mathrm{x}_1) \sim \rho(\mathrm{x}_0, \mathrm{x}_1|\xi)$, $\xi \sim \eta(\xi)$, and $z \sim \mathrm{N}(0, \mathrm{I})$.

We next show equation 17 is a solution of a stochastic differential equation as follows

$$\mathrm{d}\mathbb{E}[\mathrm{x}_t | \mathrm{x}_s = \mathrm{x}, \xi] = \mathbf{f}_t(\mathrm{x})\mathrm{d}t + \sigma_t \mathrm{dw}_t, \tag{18}$$

where $\mathbf{f}_t(\mathrm{x}) = \mathbb{E}[\dot{\alpha}_t \mathrm{x}_0 + \dot{\beta}_t \mathrm{x}_1 | \mathrm{x}_s = \mathrm{x}, \xi]$ and $\sigma_t = \sqrt{2\gamma_t \dot{\gamma}_t}$.

To prove the above argument, we proceed to verify the *drift* and *diffusion* terms respectively:

- *Drift*: It is straightforward to verify the drift $\mathbf{f}_t$ by taking the gradient of the conditional expectation $\mathbb{E}[\alpha_t \mathrm{x}_0 + \beta_t \mathrm{x}_1 | \mathrm{x}_s = \mathrm{x}, \xi]$ with respect to t.
- *Diffusion*: For the diffusion term, the proof hinges on showing $\sigma_t = \sqrt{2\gamma_t \dot{\gamma}_t}$, which boils down to prove the stochastic calculus follows that $\int_0^t \sqrt{2\gamma_s \dot{\gamma}_s} \mathrm{dw}_s = \gamma_t z$. Note that $\mathbb{E}[\int_0^t \sqrt{2\gamma_s \dot{\gamma}_s} \mathrm{dw}_s] = 0$. Invoking the Itô isometry, we have $\mathrm{Var}(\int_0^t \sqrt{2\gamma_s \dot{\gamma}_s} \mathrm{dw}_s) = \int_0^t 2\gamma_s \dot{\gamma}_s \mathrm{d}s = \int_0^t (\gamma_s{}^2)' \mathrm{d}s = \gamma_t^2$ (given $\gamma_0 = 0$). In other words, $\int_0^t \sqrt{2\gamma_s \dot{\gamma}_s} \mathrm{dw}_s$ is a normal random variable with mean 0 and variable γ_t^2, which proves that equation 17 is a solution of the stochastic differential equation 18.

Define $\Sigma_t = 2\gamma_t \dot{\gamma}_t$, we know the Fokker-Planck equation associated with equation 18 follows that

$$\begin{aligned}
0 &= \frac{\partial \rho_t}{\partial t} + \nabla \cdot \left(\mathbf{f}_t \rho_t - \frac{1}{2} \Sigma_t \nabla \rho_t \right) \\
&= \frac{\partial \rho_t}{\partial t} + \nabla \cdot \left(\left(\mathbf{f}_t - \frac{1}{2} \Sigma_t \nabla \log \rho_t \right) \rho_t \right) \\
&= \frac{\partial \rho_t}{\partial t} + \nabla \cdot \left(\left(\mathbb{E}[\dot{\alpha}_t \mathrm{x}_0 + \dot{\beta}_t \mathrm{x}_1 | \mathrm{x}_s = \mathrm{x}, \xi] - \gamma_t \dot{\gamma}_t \nabla \log \rho_t \right) \rho_t \right) \\
&= \frac{\partial \rho_t}{\partial t} + \nabla \cdot \left(b_{t|s}(\mathrm{x}, \xi) \rho_t \right),
\end{aligned} \tag{19}$$

where $b_{t|s}(\mathrm{x}|\xi) = \mathbb{E}[\dot{\alpha}_t \mathrm{x}_0 + \dot{\beta}_t \mathrm{x}_1 - \gamma_t \dot{\gamma}_t \nabla \log \rho_t | \mathrm{x}_s = \mathrm{x}, \xi]$.

Further setting $s = t$ and rewrite $b_t \equiv b_{t|t}$, we have $b_t(\mathrm{x}|\xi) = \mathbb{E}[\dot{\alpha}_t \mathrm{x}_0 + \dot{\beta}_t \mathrm{x}_1 - \gamma_t \dot{\gamma}_t \nabla \log \rho_t | \mathrm{x}_t = \mathrm{x}, \xi]$

Further define $g_t^{(i)}(\mathrm{x}|\xi) = \mathbb{E}[\mathrm{x}_i | \mathrm{x}_t = \mathrm{x}, \xi]$, where $i \in \{0, 1\}$ and $g_t^{(z)}(\mathrm{x}|\xi) = \mathbb{E}[z | \mathrm{x}_t = \mathrm{x}, \xi]$. We have that

$$\begin{aligned} b_t(\mathrm{x}|\xi) &= \mathbb{E}[\dot{\alpha}_t \mathrm{x}_0 + \dot{\beta}_t \mathrm{x}_1 - \gamma_t \dot{\gamma}_t \nabla \log \rho_t | \mathrm{x}_t = \mathrm{x}, \xi] \\ &= \dot{\alpha}_t g^{(0)} + \dot{\beta}_t g^{(1)} + \dot{\gamma}_t g^{(z)} \\ &= \mathbb{E}[\dot{\alpha}_t \mathrm{x}_0 + \dot{\beta}_t \mathrm{x}_1 + \dot{\gamma}_t z | \mathrm{x}_t = \mathrm{x}, \xi], \end{aligned}$$

where the first equality follows by equation 19 and the last one follows by taking derivative to equation 17 w.r.t. the time t.

We also observe that $\nabla \log \rho_t = -\gamma_t^{-1} \mathbb{E}[z | \mathrm{x}_t = \mathrm{x}]$.

Theorem 2 *The loss functions used for estimating the vector field follow that*

$$L_i(\hat{g}^{(i)}) = \int_0^1 \mathbb{E}[|\hat{g}^{(i)}|^2 - 2\mathrm{x}_i \cdot \hat{g}^{(i)}] \mathrm{d}t,$$

where $i \in \{0, 1, z\}$, the expectation takes over the density for $(\mathrm{x}_0, \mathrm{x}_1) \sim \rho(\mathrm{x}_0, \mathrm{x}_1|\xi)$, $\xi \sim \eta(\xi)$, and $z \sim \mathrm{N}(0, \mathrm{I})$.

Proof To show the loss is effective to estimate $g^{(0)}$, $g^{(1)}$, and $g^{(z)}$. It suffices to show

$$\begin{aligned} L_0(\hat{g}^{(0)}) &= \int_0^1 \mathbb{E}[|\hat{g}^{(0)}|^2 - 2\mathrm{x}_0 \cdot \hat{g}^{(0)}] \mathrm{d}t, \\ &= \int_0^1 \int_{\mathrm{R}^d} \left[|\hat{g}^{(0)}|^2 - 2\mathbb{E}[\mathrm{x}_0 | \mathrm{x}_t = \mathrm{x}, \xi] \cdot \hat{g}^{(0)} \right] \mathrm{d}\mathrm{x}\mathrm{d}t, \\ &= \int_0^1 \int_{\mathrm{R}^d} \left[|\hat{g}^{(0)}|^2 - 2g^{(0)} \cdot \hat{g}^{(0)} \right] \mathrm{d}\mathrm{x}\mathrm{d}t, \end{aligned}$$

where the last equality follows by definition. The unique minimizer is attainable by setting $\hat{g}^{(0)} = g^{(0)}$.

The proof of $g^{(1)}$ and $g^{(z)}$ follows a similar fashion.

C Importance Sampling

The loss functions for training the velocity and score functions are

$$\begin{aligned} \mathcal{L}_b &= \int_0^1 \mathbb{E}\Big[\frac{1}{2} \|\hat{\mathbf{b}}(s, \mathbf{x}_s)\|^2 - \big(\dot{\alpha}(s)\mathbf{x}_0 + \dot{\beta}(s)\mathbf{x}_1 + \dot{\gamma}(s)\mathbf{z} \big)^T \hat{\mathbf{b}}(s, \mathbf{x}_s) \Big] ds \\ \mathcal{L}_s &= \int_0^1 \mathbb{E}\Big[\frac{1}{2} \|\hat{\mathbf{s}}(s, \mathbf{x}_s)\|^2 + \gamma^{-1} \mathbf{z}^T \hat{\mathbf{s}}(s, \mathbf{x}_s) \Big] ds \end{aligned} \tag{20}$$

Both loss functions involve the integral over diffusion time $s \in [0, 1]$ in the form of

$$\mathcal{L} = \int_0^1 l(s) ds \approx \sum_i l(s_i), \quad s_i \sim \mathrm{Uniform}[0, 1] \tag{21}$$

However, the loss values $l(s)$ has very large variance, especially when s is near 0 or 1. Figure 3 shows an example of the distribution of $l(s)$ across multiple s. The large variance slows down the convergence of training. To overcome this issue, we apply importance sampling, similar technique used by [Song et al., 2021, Sec. 5.1], to stabilize the training. In stead of drawing diffusion time from uniform distribution, importance sampling considers,

$$\mathcal{L} = \int_0^1 l(s) ds \approx \sum_i \frac{1}{\tilde{q}(s_i)} l(s_i), \quad s_i \sim \tilde{q}(s) \tag{22}$$

Ideally, one wants to keep $\frac{1}{\tilde{q}(s_i)}l(s_i)$ as constant as possible such that the variance of the estimation is minimum. The loss value $l(s)$ is very large when s is close to 0 or 1, and $l(s)$ is relatively flat in the middle, and the domain of s is $[0, 1]$, so we choose Beta distribution $\mathrm{Beta}(s; 0.1, 0.1)$ as the proposal distribution $\tilde{q}$. As shown in Figure 3, the values of $\frac{1}{\tilde{q}(s_i)}l(s_i)$ are plotted against their s, which becomes more concentrated in a small range.

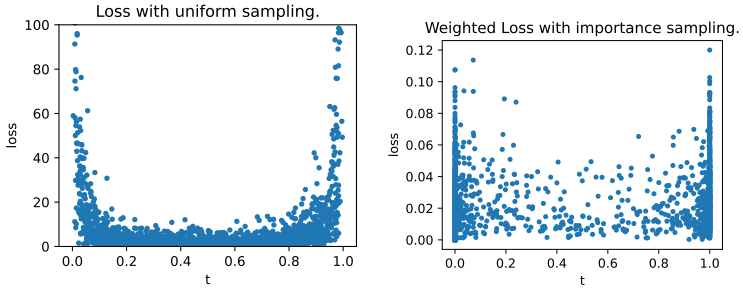

Figure 3: Comparison between uniform sampling and importance sampling. Each dot represent the loss of one sample with respect to the diffusion time.

D Experiments

D.1 Datasets and Settings

Details of the datasets are listed in Table 2.

Table 2: Properties of the datasets.

Datasets	Dimension	Frequency	Total time points	Context length
Exchange	8	Daily	6,071	30
Solar	137	Hourly	7,009	24
Traffic	963	Hourly	4,001	24
Wiki	2000	Daily	792	30

Baseline models such as DDPM, SGM, FM, and SI all involve modeling field functions, where the inputs are the state vector (in the same space of the data samples), diffusion time, and condition embedding, and the output is the generated sample. The field functions correspond to the noise prediction function in DDPM; the score function in SGM; the vector field in FM; and the velocity and score functions in SI. To make a fair comparison between these models, we use the same neural networks for these models. In particular, DDPM and SGM-based models can only generate samples by transporting Gaussian noise distribution to data distribution. So we modify the framework by replacing the context time point $\mathbf{x}_t$ with Gaussian noise. Flow matching can easily fit into this framework by replacing the denoising objective with the flow matching objective. The modified framework is shown in Figure 1. We model the map from the previous time series observation to the next (forecasted) value. We argue this is a more natural choice than mapping from noise for each time series prediction step. Recent Vec-LSTM from [Salinas et al., 2019] is compared as a pure recurrent neural network model whose probabilistic layer is a multivariate Gaussian.

D.2 Metrics and Preprocessing

The probabilistic forecasting is evaluated by Continuous Ranked Probability Score (CRPS-sum) Koochali et al. [2022], normalized root mean square error via the median of the samples (NRMSE), and point-metrics normalized deviance (ND). The metrics calculation is provided by `gluonts` package Alexandrov et al. [2019] by calling module `gluonts.evaluation.MultivariateEvaluator`. We follow the preprocessing steps as in [Salinas et al., 2019]. In all of the cases smaller values indicate better performance.

The RNN for the history encoder has 1 layer and 128 latent dimension; The field function is modeled with Unet-like structure Ronneberger et al. [2015] with 8 residual blocks, and each block has 64 dimensions. To stabilize the training, we also use paired-sampling for the stochastic interpolants introduced by [Albergo et al., 2023a, Appendix C].

$$\begin{aligned}\mathbf{x}_s =&\alpha(s)\mathbf{x}_0 + \beta(s)\mathbf{x}_1 + \gamma(s)\mathbf{z}\\ \mathbf{x}'_s =&\alpha(s)\mathbf{x}_0 + \beta(s)\mathbf{x}_1 + \gamma(s)(-\mathbf{z})\\ &s \in [0,1], \mathbf{z} \sim \mathcal{N}(\mathbf{0}, I)\end{aligned}$$

The baseline models are trained with 200 epochs and 64 batch size with learning rate 10^{-3}. The SI model is trained with 100 epochs and 128 batch size with learning rate 10^{-4}. We find if the learning rate is too large, SI may not converge properly.

D.3 Additional Results

Additional forecasting results using ND-sum and NRMSE-sum have also been presented in the tables as follows.

	Exchange rate	Solar	Traffic	Wiki
DDPM	0.011±0.004	0.377±0.061	0.064±0.014	0.093±0.023
FM	0.011±0.001	0.445±0.031	0.041±0.002	80.624±89.804
SGM	0.01±0.002	0.388±0.026	0.08±0.053	0.122±0.026
SI	0.008±0.002	0.399±0.065	0.089±0.006	0.091±0.011

Table 3: ND-sum. A smaller number indicates better performance.

	Exchange rate	Solar	Traffic	Wiki
DDPM	0.013±0.005	0.72±0.08	0.094±0.029	0.123±0.026
FM	0.014±0.002	0.849±0.072	0.059±0.007	165.128±147.682
SGM	0.019±0.004	0.76±0.066	0.109±0.064	0.164±0.03
SI	0.01±0.003	0.722±0.132	0.127±0.003	0.117±0.011

Table 4: NRMSE-sum. A smaller number indicates better performance.